REMARKS

Favorable reconsideration of this application in view of the above amendments and the following remarks is respectfully requested. By this amendment, claims 1, 3-5, 8, 10, 11, and 24-28 have been amended without prejudice or disclaimer. Currently, claims 1-28 are pending in the application.

Interview with the Examiner

The Examiner is thanked for conducting a personal interview with two of the applicants (Michael Hinchey and James Rash) and applicants' representative (Heather Goo, Reg. No. 37,336) on November 10, 2008. While no specific agreement was reached as to amendments to the claims or the rejections of the claims on the cited art during the interview, Examiner Deng clarified her position and listened attentively to Applicants' presentation and questions. Examiner Deng did indicate or suggest claim language to more clearly claim the subject invention. Applicants' have taken the Examiner's suggestions and incorporated them into the claims, as amended.

Rejection Under 35 U.S.C. §102(b)

Claims 1-4, 10-23, and 26-28 were rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 6,289,502 to Garland et al. (hereinafter, "Garland '502"). This rejection is respectfully traversed.

The subject invention relates to providing a method such that user level requirements described in a natural language, that is, an informal specification or requirements specification, are translated to a mathematical specification, that is, a formal specification. The process described includes a series of iterations whereby the natural language, which is a vocabulary of user level requirements, is translated into traces and logic is inferred to

provide a formal design specification. In particular, a system and method for deriving a process-based specification for a system is disclosed. The formal, process-based specification is mathematically inferred from an informal, trace-based specification. The trace-based specification is derived from a non-empty set of traces or natural language scenarios. The process-based specification is mathematically equivalent to the trace-based specification. Code is generated, if applicable, from the process-based specification. A process, or phases of a process, using the features disclosed can be reversed and repeated to allow for an interactive development and modification of legacy systems. The process is applicable to any class of system, including, but not limited to, biological and physical systems, electrical and electro-mechanical systems in addition to software, hardware, and hybrid hardware-software systems.

As explained in more detail in the specification, as filed, FIG. 1 illustrates an exemplary software development system 100. Software development system 100 includes a data flow and processing points for the data, and mechanically converts different types of specifications (either natural language scenarios or descriptions, or trace specifications, which are effectively pre-processed scenarios) into process-based formal specifications on which model checking and other mathematics-based verifications are performed, and then optionally converts the formal specification into code.

Generally, a natural language scenario 110 is a scenario in natural language text that describes the system's actions in response to incoming data and the internal goals of the system. The set of natural language scenarios 110 is constructed in terms of individual scenarios written in a structured natural language. Different scenarios may be written by different stakeholders of the system, corresponding to the different views they have of

how the system will perform. Natural language scenarios 110 may be generated by a user with or without mechanical or computer aid.

Context sensitive editor 120 takes natural language scenarios 110 and database of domain parameters 115, and produces a trace specification 130 based on the inputted scenarios. Trace specification 130 correlates to natural language scenarios 110. In formal methods, and other areas of software engineering, a trace is a sequence of events that a process has engaged in up to a given point in time. Trace specification 130 includes traces that are a list of all possible orderings of actions as described by the developers that are taking place within the scenarios. More specifically, traces are lists of computational actions. Traces list the actions in a sequential manner.

Code 195 preferably comprises a sequence of instructions to be executed as a software system. Code 195 is an accurate description of the system defined by natural language scenarios 110 that is mathematically provable. Code 195 is used for verification 196 of the natural language scenarios 110 as well as trace specification 130. Further, code 195 may be used as a basis for model checking 197. Code 195 comprises executable computer code, which may be translated before execution.

As summarized and shown in the flowchart of FIG. 2, natural language scenarios are translated to a trace-based specification; next, a process-based specification is derived from the trace-based specification; and then, code is developed from the process-based specification.

Claims 1-4 and 10-23

Independent claim 1 describes a method for deriving a process-based specification includes deriving a trace-based specification from a non-empty set of traces by a processor, and mathematically inferring that the process-

based specification. A trace is a sequence of actions expressed as strings representing a history of an execution of a process. Mathematically inferring includes applying the Laws of Concurrency in reverse to a set of system traces to determine the process-based specification. The process-based specification is mathematically equivalent to the trace-based specification. The Laws of Concurrency are algebraic laws that (a) allow at least one process to be manipulated and analyzed; (b) permit formal reasoning about equivalences between processes; and (c) determine traces from the at least one process.

Garland '502 relates to taking a formal design specification, that is, a system specification, and validating the design specification. (*See* col. 2, lines 7-15). In Garland '502, the design specification is provided in a formal language, i.e., IO Automata. From this input, i.e., the formal design or system specification, code can be developed. (*See* col. 13, lines 58-65). Further, as the Examiner points out, "[T]his form [that is, the system specification in the IO Automata form,] is a mathematical description of the underlying automaton." *Id.* at lines 61-62. That is what comprises a system specification or formal design specification.

Therefore, the system specification of Garland '502 is the same as the process-based specification that is derived in the subject claim. As explained in the specification, for example, at page 4, paragraph 13 ("software development system 100 [] converts [] natural language scenarios or [] trace specifications [] into process-based formal specifications on which model checking and other mathematics-based verifications are performed, and then optionally converts the formal specification into code"), the process-based specification is a formal specification. However, the instant claim is for a method for deriving a process-based specification. The claimed method

starts with a trace-based specification and results in a process-based specification.

Thus, Garland '502 fails to teach or describe deriving a process-based specification includes deriving a trace-based specification from a non-empty set of traces, and mathematically inferring that the process-based specification is mathematically equivalent to the trace-based specification, as recited in independent claim 1. Rather, Garland '502 starts with the formal, process-based specification, validates this formal specification, and then converts the formal specification to code. There is no translation from natural language scenarios to trace-based specification to deriving a process-based specification in Garland '502.

Therefore, the subject matter of independent claim 1 is not taught or described by Garland '502. Thus, independent claim 1 is allowable over the cited art of Garland '502. Further, claims 2-4 and 10-23, which depend from independent claim 1, are also not taught or described by Garland '502, and are allowable over Garland '502.

Accordingly, withdrawal of this rejection as to independent claim 1, and claims 2-4 and 10-23, which depend directly or indirectly therefrom, is respectfully requested.

Claim 26

Independent claim 26, as amended, describes a system adapted for deriving a process-based specification that includes at least one natural language scenario, a computer-readable medium having instructions stored thereon for deriving a trace-based specification from the at least one natural language scenario, and an inference engine to mathematically infer the process-based specification from the trace-based specification. Mathematically inferring includes applying the Laws of Concurrency in

reverse to a set of system traces to determine the process-based specification. The process-based specification is mathematically equivalent to the trace-based specification. The Laws of Concurrency are algebraic laws that (a) allow at least one process to be manipulated and analyzed; (b) permit formal reasoning about equivalences between processes; and (c) determine traces from the at least one process.

For at least the reasons above, Garland '502, however, fails to teach or describe a system adapted for deriving a process-based specification. Additionally, Garland '502 lacks any teaching of at least one natural language scenario, a computer-readable medium having instructions stored thereon for deriving a trace-based specification from the at least one natural language scenario, and an inference engine to mathematically infer the process-based specification from the trace-based specification, as recited in independent claim 26.

Rather, as pointed out above, Garland '502 starts with the formal, process-based specification, validates this formal specification, and then develops code from the formal specification. In Garland '502, there is no derivation of a trace-based specification from a natural language scenario or mathematical inference of a process-based specification from the trace-based specification, as claimed. Garland '502 starts with the formal, process-based specification, which is **the result of the instant claim.**

Therefore, the subject matter of amended independent claim 26 is neither taught nor described by Garland '502. Thus, independent claim 26 is allowable over Garland '502.

Accordingly, withdrawal of this rejection as to independent claim 26 is respectfully requested.

Claim 27

Independent claim 27, as amended, describes another system adapted for deriving a process-based specification for use in a system that includes a non-empty set of traces, a computer-readable medium having instructions stored thereon for deriving a trace-based specification from the set of traces, and an inference engine to mathematically infer the process-based specification from the trace-based specification. A trace is a sequence of actions expressed as strings representing a history of an execution of a process. Mathematically inferring includes applying the Laws of Concurrency in reverse to a set of system traces to determine the process-based specification. The process-based specification is mathematically equivalent to the trace-based specification. The Laws of Concurrency are algebraic laws that (a) allow at least one process to be manipulated and analyzed; (b) permit formal reasoning about equivalences between processes; and (c) determine traces from the at least one process.

Again as noted previously, Garland '502 fails to teach or describe a system adapted for deriving a process-based specification. Likewise, Garland '502 lacks any teaching of a non-empty set of traces, a computer-readable medium having instructions stored thereon for deriving a trace-based specification from the set of traces, and an inference engine to mathematically infer that the process-based specification is mathematically equivalent to the trace-based specification, as recited in independent claim 27.

Again, Garland '502 starts with the formal, process-based specification, validates this formal specification, and then develops code from the formal specification. In Garland '502, there is no derivation of a trace-based specification from a set of traces or mathematically inferring a process-based specification from the trace-based specification. Garland '502

starts with the formal, process-based specification, which is the result of the instant claim.

Therefore, the subject matter of independent claim 27 is not taught or described by Garland '502. Thus, independent claim 27 is allowable over Garland '502.

Accordingly, withdrawal of this rejection as to independent claim 27 is respectfully requested.

Claim 28

Independent claim 28, as amended, describes another method for deriving a process-based specification for a system that performs actions. The method includes receiving at least one natural language scenario describing the actions, generating a trace-based specification from the at least one natural language scenario, and mathematically inferring the process-based specification from the trace-based specification. Mathematically inferring includes applying the Laws of Concurrency in reverse to a set of system traces to determine the process-based specification. The process-based specification is mathematically equivalent to the actions defined above. The Laws of Concurrency are algebraic laws that (a) allow at least one process to be manipulated and analyzed; (b) permit formal reasoning about equivalences between processes; and (c) determine traces from the at least one process.

As to claim 28, Garland '502 fails to teach or describe receiving at least one natural language scenario describing the actions, generating a trace-based specification from the at least one natural language scenario, and mathematically inferring the process-based specification from the trace-based specification, as recited. Garland '502 starts with the formal, process-based specification, validates this formal specification, and then develops

code from the formal specification. There is no claimed receiving at least one natural language scenario describing the actions, no claimed generating a trace-based specification from the at least one natural language scenario, and no claimed mathematically inferring the process-based specification from the trace-based specification. Garland '502 starts with the formal, process-based specification, which is the **result of the instant claim**.

Therefore, the subject matter of independent claim 28 is not taught or described by Garland '502. Thus, independent claim 28 is allowable over the cited art of Garland '502. Accordingly, withdrawal of this rejection as to independent claim 28 is respectfully requested.

Rejection Under 35 U.S.C. §103(a)

Claims 5-9, 24, and 25 were rejected under 35 U.S.C. §103(a) as being unpatentable over Garland '502 in view of U.S. Patent No. 6,405,364 to Bowman-Amuah (hereinafter, "Bowman '364"). This rejection is respectfully traversed.

As presented above, the subject invention generally relates to providing a method such that user level requirements described in a natural language, that is, an informal specification or requirements specification, are translated to a mathematical specification, that is, a formal specification. The process described includes a series of iterations whereby the natural language, which is a vocabulary of user level requirements, is translated into traces and logic is inferred to provide a formal design specification.

Claim 5 describes that in the method of claim 4, the Laws of Concurrency are reversed by embedding the Laws of Concurrency in the inference engine. Claim 6 further describes that the embedding is syntactic or shallow. Claim 7 further describes that the embedding is semantic or deep.

Claim 8 recites describes that in the method of claim 4, the Laws of Concurrency are reversed so that an equivalent process expression is output in response to a given input of at least one trace. Claim 9 further describes that multiple process expressions are given as output in response to inputs of the at least one trace.

Claim 24, as amended, describes that the method of claim 1 further includes reverse engineering an existing system using the deriving step and the inferring step. Reverse engineering includes analyzing a process-based specification and applying the Laws of Concurrency using an inference engine having domain knowledge to infer a set of traces equivalent to the process-based specification and to derive a set of scenarios for the system equivalent to the process-based specification. The scenarios are natural language text that describes the system's actions in response to incoming data and the internal goals of the system.

Claim 25, as amended, describes that the method of claim 1 further includes reverse engineering an existing system back to a set of traces using the deriving step and the inferring step. Reverse engineering includes analyzing a process-based specification and applying the Laws of Concurrency using an inference engine having domain knowledge to infer a set of traces equivalent to the process-based specification and to derive a set of scenarios for the system equivalent to the process-based specification. The scenarios are natural language text that describes the system's actions in response to incoming data and the internal goals of the system.

Garland

As presented above, Garland '502 relates to taking a formal design specification, that is, a system specification, and validating the design specification. In Garland '502, the design specification is provided in a

formal language, i.e., IO Automata. From this input, i.e., the design specification, code can be developed.

For at least the reasons presented above, Garland '502 fails to teach or describe the subject matter of claims 5-9, 24, and 25, which depend directly or indirectly from allowable independent claim 1. Therefore, claims 5-9, 24, and 25 are also allowable over Garland '502.

Bowman

Bowman '364 relates to system building techniques in a development architecture framework. Bowman '364 describes various tools that can be used to specify the requirements of the system to be built and an implementation strategy, or improve the performance and maintenance of a system. Bowman '364 assists in organization of a development activity according to a developmental framework concept that is dependent upon having a pre-existing system model as a starting point. Bowman '364 does not relate to any specific method for applying the laws of concurrent programming or any other formal modeling technique in transforming original user requirements into a formal model.

Bowman '364, like Garland '502, relates to formal, design specifications, that is, a pre-existing system model, not natural language scenarios or a trace-based specification as in the claimed invention.

The Examiner believes that because Bowman '364 mentions the process of reverse engineering, this teaches or suggests the claimed features. However, even *arguendo* if the Examiner's position is correct (which Applicants contend is not the case), Bowman fails to teach or suggest the claimed subject matter. As to claims 5-7, there is no teaching or suggestion that the Laws of Concurrency are reversed by embedding the Laws of Concurrency in the inference engine. Likewise, as to claims 8 and 9, there is

no teaching or suggestion that for at least one input trace, there is one or multiple process expressions output.

As to claim 24, Bowman '364 fails to teach or suggest the deriving step or the inferring step.

Similarly as to claim 25, Bowman '364 also fails to teach or suggest reverse engineering back to a set of traces.

Therefore, Bowman '364 fails to overcome the above-noted deficiencies in Garland '502, and thus, even in combination with Garland '502, fails to teach or describe the subject matter of claims 5-9, 24, and 25, which depend directly or indirectly from allowable independent claim 1.

Accordingly, withdrawal of this rejection as to claims 5-9, 24, and 25 is respectfully requested.

Conclusion

Applicants submit that all pending claims in this application, claims 1-28, are in condition for allowance, and formal notice of such is solicited. In the event that the examiner has any questions, she is respectfully requested to contact the undersigned at the number listed below.

The Commissioner is authorized to charge any additional fees or credit any overpayment to Deposit Account No. 14-0116.

Respectfully submitted,

January 27, 2009

By: /Heather Goo/
Heather Goo
Registration No. 37,336

NASA
Goddard Space Flight Center
Office of Patent Counsel
8800 Greenbelt Road
Greenbelt, MD 20771
301-286-0602 (direct)
301-286-7351 (main)